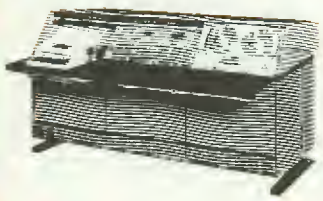
SP-149A

An Introduction to

PB440 MICROPROGRAMMING



pb Packard Bell Computer

An Introduction to

PB440 MICROPROGRAMMING

FOREWORD

This Microprogramming Manual is the first in a series of technical and application reference manuals on the Packard Bell PB440 Computer. Others to be made available include programming manuals on command sets, a Fortran Programming Manual and technical manuals on peripheral equipment.

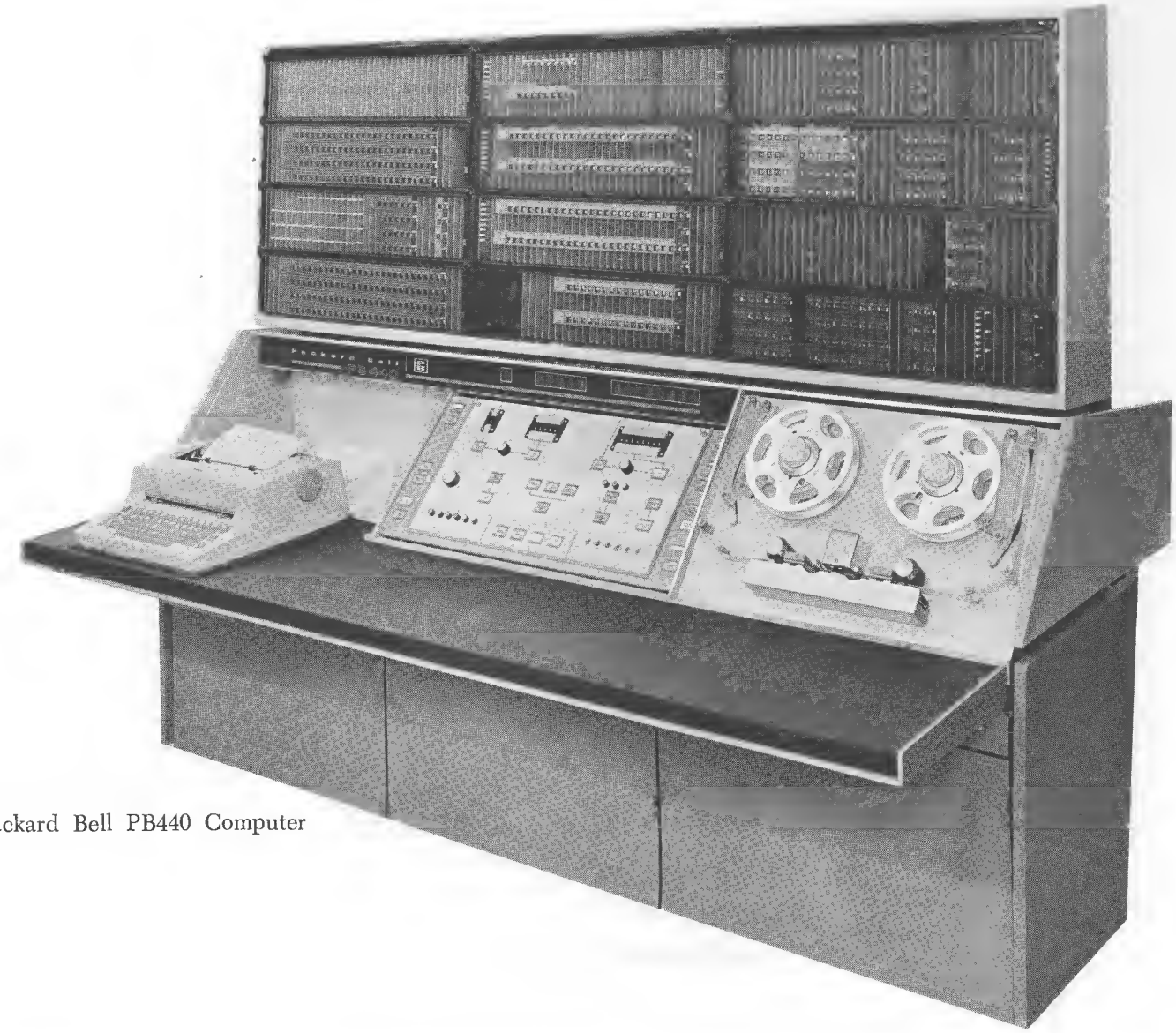
Although this present manual covers microprogramming procedures in some detail, few PB440 users will wish to program their machine at that level. However, a study of microprogramming techniques will be helpful in disclosing many of the more significant details of the PB440 hardware. To facilitate use of the PB440, representative command lists covering engineering scientific computation and realtime systems computing and control will be provided with the machine.

REVISION NOTICE

This manual is a revised edition of "An Introduction to the PB440 Computer," dated November 1962. All information in this revised edition supersedes the information contained in the November 1962 edition.

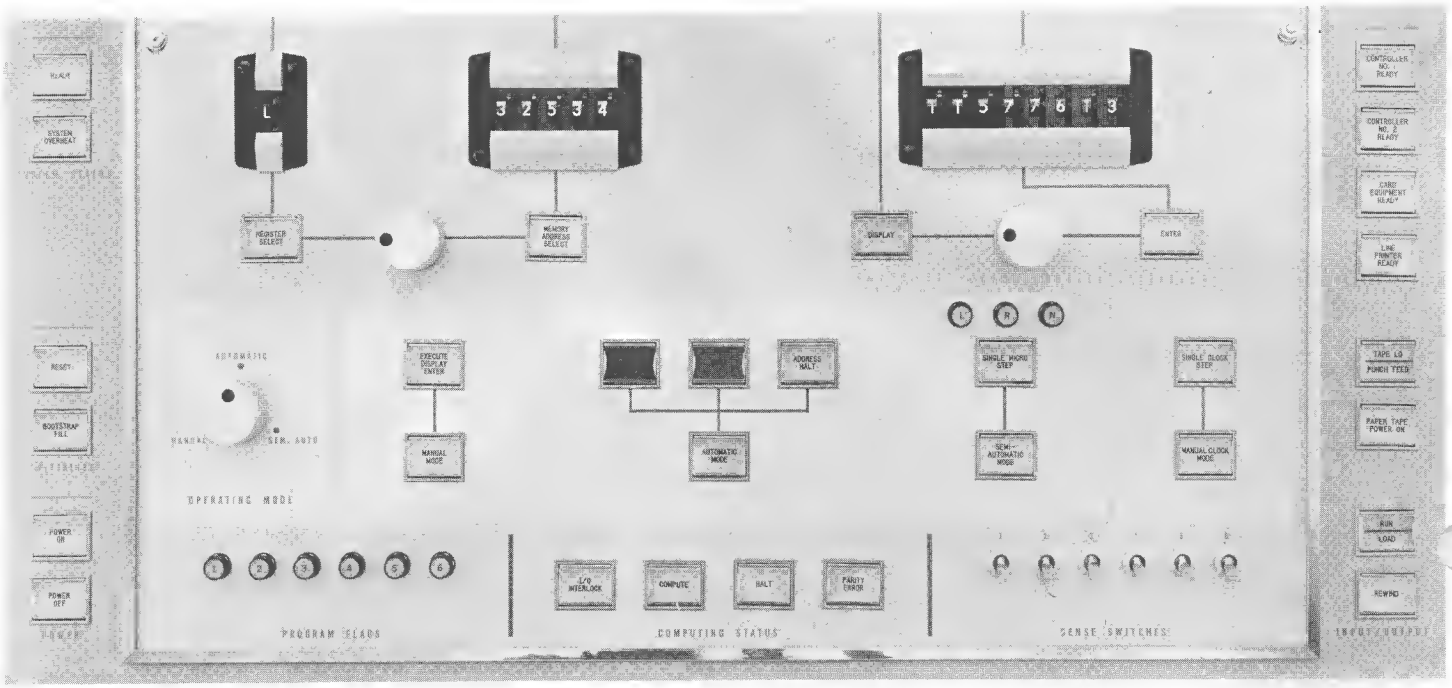
TABLE OF CONTENTS

ILLUSTRATIONS



Packard Bell PB440 Computer

PB440 Computer Control Console



GENERAL DESCRIPTION

The PB440 Computer is a core-memory computer whose basic operations can be performed at the clock rate of the computer. The programmer has complete control over the specific operation performed during nearly every clock pulse; in this sense the computer is said to be micro-programmed. Sequences of basic operations can be stored in memory and executed much like a conventional subroutine, so that operations normally considered to be "commands" on a conventional computer can be described in terms of their elementary operations, and can be changed at will. In this sense the computer is said to be of the stored-logic type.

Two separate types of core memory are provided, although the memory is addressed as a single homogeneous block. Main memory, comprising the bulk of the storage capacity of the machine, consists of magnetic cores from which one word of information can be obtained in 2 microseconds. The memory cycle time is 5 microseconds. Fast memory, designed to hold the sequences of elementary steps which define an "instruction", consists of a set of BIAX cores, from which a word can be obtained in less than 1 microsecond. Since these cores are read non-destructively, the term "cycle time" does not apply to fast memory. The word length of both memories is 24 binary digits (bits). (Internal parity checking is provided automatically, and utilizes an additional bit in each word for this purpose. This bit is not available for programming purposes.)

The basic machine contains 4096 words of main memory and 256 words of fast memory (see Figure 1). Main memory can be expanded to a maximum of 28,672 words, and fast memory can be expanded (in 256 word blocks) to 4096 words for a maximum of 32,768 words of directly addressable memory.

The input-output equipment planned for the basic computer consists of a photo electric paper-tape reader capable of reading 500 characters per second, a paper-tape punch capable of punching at a rate of 110 characters per second, and an electric typewriter which can print 16 characters per second. Additional input-output equipment, consisting of IBM-compatible card and magnetic tape devices, is available at extra cost. A high-speed line printer will also be available as optional equipment.

THE INSTRUCTION REPERTOIRE

The PB440 does not have a set of instructions in the same sense that conventional computers have. Instead, its basic design provides the programmer with the option of describing, in terms of elementary operations, a sequence of steps which define an instruction. Nearly every clock pulse is available to the programmer for this purpose. A complete sequence of elementary steps which describe an operation (instruction) in the conventional sense is called a *microutine*. To avoid confusion we will refer to the instruction described in this way as a "macro-instruction". A complete instruction set consists of a control sequence (to determine which of the various macro-instructions is indicated) and a set of microutines, each of which defines a macro-instruction. This instruction set is normally stored in fast memory, although micro-steps can be executed from main memory (at a slower rate) if so desired, or if there is not enough room in fast memory to contain them all.

There are, of course, many different ways in which one could describe the individual logic steps which constitute a conventional computer instruction. The logic designer usually describes them with a set of Boolean

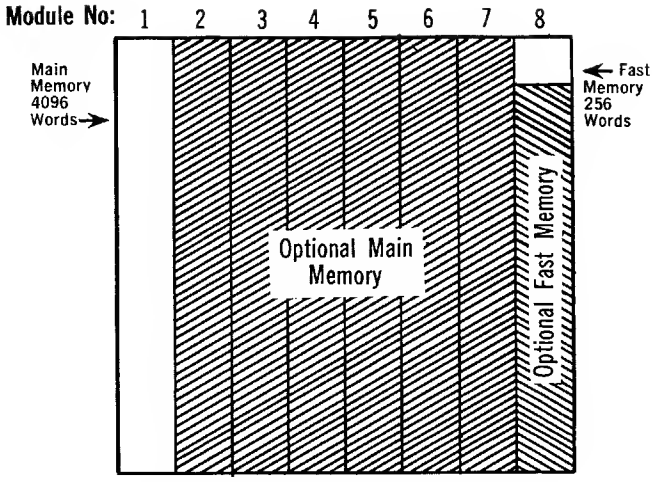


Figure 1. PB440 Overall Memory Layout

Each module can contain 4096 words, 24 bits in length.

Main memory access time:	2μsec
Main memory cycle time:	5μsec
Fast memory access time:	<1μsec

Memory Addressing Scheme

Module No.	Octal Addresses	Decimal Addresses
1	00000 to 07777	00000 to 04095
2	10000 to 17777	04096 to 08191
3	20000 to 27777	08192 to 12287
4	30000 to 37777	12288 to 16383
5	40000 to 47777	16384 to 20479
6	50000 to 57777	20480 to 24575
7	60000 to 67777	24576 to 28671
8	70000 to 77777	28672 to 32767
1st Block of Fast Memory	70000 to 70377	28672 to 28927

equations, each equation consisting of many terms of differing meanings. He uses the same technique to describe the "control" section of the computer. Since, to a large extent at least, the design of the macro-instructions for the PB440 computer has been left to the programmer, a method of description was chosen which is already familiar to him; that is, in terms of a sequential series of elementary steps which he calls a program. It should be emphasized that there is a very real difference between a microutine, which describes a macro-instruction, and a computer program, which consists of a series of macro-instructions designed to solve a problem, even though the methods used to obtain the two are very much alike.

DESIGNING AN INSTRUCTION SET

The enormous advantage of being able to choose an instruction set to suit a particular problem is probably obvious to anyone who has written a program for a digital computer. The programmer can, in a very real sense, design the computer to fit his problem. The resulting program which he can write will be far shorter and more efficient than if he were restricted to an instruction set not of his own choosing, and he need not have a group of instructions available which he finds he cannot use. As an example, by designing an instruction set to execute a program written originally in the FORTRAN language and compiled automatically, it was found that the resulting program (generated by the compiler) would be about half as long as one generated from the same source program on a more conventional computer. This saving, in terms of the length of the program, is effected by having available instructions designed to concisely describe operations available in the FORTRAN source language. In the same way, the compiler itself can be made compact and efficient by writing it in an instruction set which can describe, in a very few instructions, the manipulations which a compiler must perform to translate a source program into a machine language program.

It should be recognized that not all programmers will want to be bothered with the problems of computer design every time they write a program. They need not be. Packard Bell recognizes its responsibility in this regard, and will provide several complete instruction sets with the computer upon delivery, in addition to the FORTRAN system now under construction. Assembly routines and other "utility" routines will be provided which are compatible with each of the instruction sets, and provision will be made for the programmer who may want to augment an instruction set with a few instructions of his own design.

The design of an instruction set may be conveniently divided into three parts:

1. A macro-instruction format is chosen, usually with an eye toward rapid interpretation. This format need not be restricted to a word length of 24 bits, nor to single address instructions. The instructions need not refer to an "accumulator" unless this is useful, and more than one "accumulator" may be referenced if desired. Conventional or unconventional indexing may be described. Any other useful features, such as indirect addressing, relative-indirect addressing, or whatever special functions are needed, can be included.
2. The control sequence is described. It must be capable of distinguishing between the macro-instruction formats chosen (if there were more than one) and will normally provide operations common to most, or all, of the instructions such as indexing and indirect addressing. It will normally obtain the parameters, if any, needed for the execution of the instruction.
3. The microutines, each of which defines the execution of an instruction, are written. These may be of any length and complexity desired, and may call on, or transfer control to, other microutines. Astute design of the microutines can shorten the whole instruction set considerably, and provide many useful combinations not normally found on conventional computers. As an example, assume we need a fixed-point add instruction, and we also require an instruction which stores the content of an "accumulator" into memory. Very little is required to provide an additional instruction which performs both of these functions in sequence. This new instruction becomes "add the content of memory to the accumulator, and store the result back in the same place in memory". The new instruction can make use of the other two microutines already present, at little cost in space or in execution time, and provides the facility for increasing the value of a "counter" in memory. For example, using only two instructions:

CLA	1	Clear and add "1" to the accumulator
ADS	CNTR	Add "CNTR" and store the sum in "CNTR"

It should be remembered that the amount of time required to interpret the macro-instruction should be kept as short as possible, since this amount of time must be added to the execution time of each macro-instruction, no matter how fast that execution may be. It is an "overhead" which must be included when considering the overall execution speed of any particular instruction set.

This "overhead" time can be minimized in two ways. Many of the micro-steps available to the programmer are designed to make interpretation easy and fast, and the programmer should become familiar with them before he attempts to design an instruction set. He should also choose the individual macro-instructions with care so that as few of them as possible are needed to do his job.

THE PB440'S INTERNAL SEQUENCE

The registers available in the PB440 Computer are shown in Figure 2. For the moment, we need consider only the registers labeled "P" and "E".

The individual micro-steps are stored in memory in pairs (see Fig. 3) and are executed in sequence from left to right. The "P" register functions as a micro-pair location counter, and normally contains the address of the next micro-pair to be executed. It is automatically incremented by 1 each time a micro-pair is obtained from memory for execution.

The "E" register holds the pair of micro-steps being executed, and may be thought of as the micro-level command register. If we examine the automatic sequence of operations, beginning at the moment the micro-pair is inserted into the "E" register for execution, we would observe the following operations:

1. The "P" register is incremented by 1.
2. The left-hand micro-step is examined by the computer logic and the indicated micro-step operation is performed.
3. At the same time, the right-hand micro-step is examined to see if it involves a memory reference, or indicates the "P" register is to be changed. For the moment, assume it does not.
4. Following execution of the left-hand micro-step the right-hand micro-step is examined and executed as before.
5. During the execution of the right-hand micro-step, the next micro-pair is inserted into the "E" register and the cycle repeats.

Normal timing for this execution sequence, assuming the next pair is located in fast memory, is 2 clock pulses. To a first approximation one can say that execution proceeds at a rate of 1 micro-step per clock pulse. When the right-hand micro-step changes the content of the "P" register, however, the proper micro-pair cannot be obtained from memory until this operation is completed. This condition requires that the computer wait for one additional clock pulse, so that 3 clock pulses are required to execute the micro-pair.

The memory-bus structure allows either fast or main memory to be referenced during any clock pulse, but not both at the same time. Therefore, if the right-hand micro-step is one which references either main or fast memory, the next micro-pair cannot be obtained at the same time, and the computer must wait for 1 additional clock pulse after the memory access. There is no compounding of waiting periods for both changing "P" and accessing memory in a right-hand micro-step.

Since transfer of control is effected by altering the content of the "P" register (which is addressable), actual control transfer will not occur until *after* execution of the right-half micro-step. In general, then, it can be seen that contol transfer micro-steps and memory reference micro-steps should be kept in the left-half of the pair wherever possible.

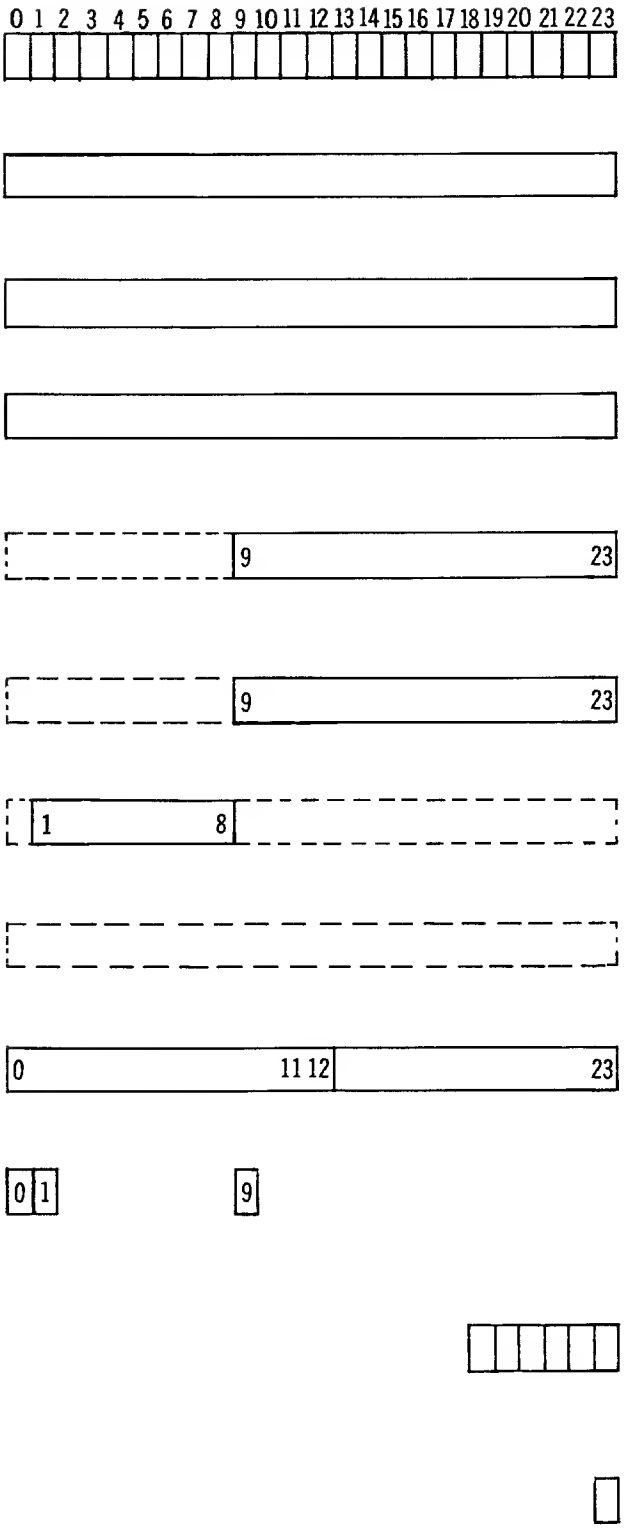
One restriction is inherent in the micro-pair structure. Since the "P" register contains the location of each *pair*, it is possible to transfer control only to a left-half micro-step. While this restriction might indicate that many "no-op" micro-steps would be required, programmer ingenuity can keep them to a minimum.

One consequence of the micro-pair structure should be mentioned. Conditional-test micro-steps exhibit a different response according to whether they are located in the left- or the right-hand side of a micro-pair. If the conditional test micro-step is located in the left-half of a micro-pair and if the condition being tested is met, the right-hand micro-step is executed as written. If the condition is not met, the right-hand micro-step is not executed. However, if the conditional-test micro-step is in the right-half position, the next complete micro-pair will be executed as written if the condition is met, or it will be skipped if the condition is not met.

THE REGISTERS

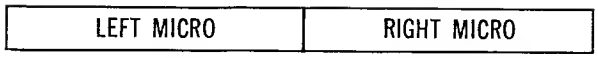
The working registers are shown in diagram form in Figure 2. All of the registers shown are directly addressable with the exception of E, the micro-pair command register. The dotted lines indicate portions of a register which are not actually present in hardware; reference to such a region has the same effect as if the register contained all zeroes in those bit positions. The Q register is all empty; it may be considered as a register permanently containing zeroes.

The working registers consist of the A, B, C, and D registers of 24 bits each, the N register consisting of 8 bits, the L register (15 bits in length), three carry toggles, 6 "program flags" (which are addressable 1 bit registers), and a parity toggle. Not shown in Figure 2 are those elements which are available on the computer console (6 sense-switch toggles, address switches representing a 15 bit address, and another set of switches representing a full 24 bit word). These elements can be referenced by the programmer (as can various input-



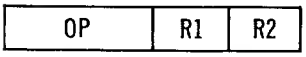
Reg. Name	Reg. Number	Use
A	1	Arithmetic Register
B	2	Arithmetic Register
C	3	Arithmetic Register
D	4	Arithmetic Register and Macro-Instruction Interpretation Register
L	5	Macro-Program Location Counter or General Address Register
P	7	Logic Level Location Counter
N	6	Repeat Count Register or Exponent Register
Q	0	Phantom Register
E	–	Micro-Pair Command Register (Not Addressable)
CARRY TOGGLES		Indicate Carry from Bit Positions 0, 1, and 9
PROGRAM FLAGS		Boolean (1 Bit) Registers
PARITY TOGGLE		Indicates parity of the last memory access micro-step

Figure 2. PB440 Register Configuration



Micro-steps are stored in pairs. Each micro-step occupies 12 bit positions.

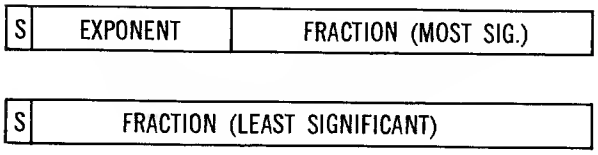
For some micro-steps the right-hand 6-bit field is treated as a unit, and is called the modifier field.



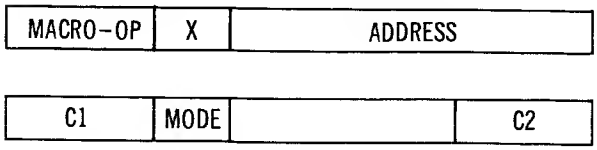
For other micro-steps the modifier field is used in two parts, R1 and R2. Each of these can address one register.

REGISTER FORMATS FOR INSTRUCTION INTERPRETATION AND ARITHMETIC

One data-word format consists of the sign position (bit 0) and magnitude (bit positions 1-23).



Floating-point data require two words. The sign position of the most significant word indicates the sign of the complete word. The other sign position is unused. The exponent field (8 bits) is always positive; excess–128 exponent convention is used to allow both positive and negative exponents.



The D register allows speeial formats. One possible format for macro-instructions consists of an op-code, 7 index registers and an address. Special fields (like C1 and C2) can be used to address memory, and the bits of the mode field can be tested individually.

Many miero-steps operate on the full 24-bit word, called a logical word.

Figure 3. PB440 Word Formats

output elements and devices) but are not considered in detail in this discussion of the registers.

In general, it is expected that most arithmetic and logical operations will be performed utilizing the A, B, C, and D registers. It is expected that the L register will be used as a macro-instruction location counter, though it should be emphasized that the design in no way restricts the use of any register (except, of course, for "P" and "E") to any particular function. Several micro-step operations reference specific registers, but, for the most part, the programmer is free to decide the function of the registers for himself.

The carry toggles are set by certain micro-steps automatically, and are designed to indicate carry-out from bit positions 0, 1, and 9. That is, a carry toggle is made true (or "1") if carry-out occurs from its bit position, or made false if no carry-out occurs. They may be set and tested individually as well. The 6 program flags may be set and tested at the discretion of the programmer, and their content, as well as the content of any other register, may be displayed on the console. The parity toggle contains the parity of the last executed memory access micro-step, and it may be referenced by the programmer.

The N register serves as a repeat counter for those micro-steps which are of a repetitive nature, such as multiplication, division, and shifting operations. It may also be used for other purposes when desired.

Figure 3 shows various word formats in diagram form. Various fields in the word have been labeled so that they may be referred to during the description of the individual micro-steps. It will be noted that the N register is located so that it can hold the "exponent" field of a floating-point format word. Also, the "fraction" portion of a floating-point format word consists of 15 bits and can also contain an address for reference to the memory system of the computer.

Several micro-steps reference only the D register which can be used to good effect as a macro-instruction command register. Any register may be shifted, but the N register should be shifted only after careful consideration of the net effect.

THE MICRO-STEPS

Figure 3 shows two possible formats for a micro-step. While many micro-steps utilize both the R1 and R2 fields to designate one of the registers, not all of them do so, and these 3 bit fields may have special meaning for certain of the micro-steps. In some cases the full 6 bit field, exclusive of the operation field, is used as a unit and reference to the register (or registers) is implied by the operation field.

In general, where the R1 and R2 fields are used to designate two registers, both fields may designate the same register. In some cases it is not meaningful to do this since no change results, but it is always permitted. For convenience, the individual micro-steps have been grouped in the descriptions below.

Memory Access Operations

Memory accesses may refer to either main memory or to fast memory. "Load" micro-steps retrieve a full 24 bit word from memory, and "Store" operations place the full 24 bit word into the designated memory cell. Where a register is less than 24 bits in length the "missing" portion will always store zeroes into memory in those bit positions.

When main memory is referenced for a load operation, 2 clock pulses are required to read the content of the memory cell into the designated register. Following the main memory access, 3 clock pulses are required to establish (or re-establish) the content of the memory cell. This time may be used for useful computing, and is referred to as "shadow time". Should a reference to main memory be made during this three clock-pulse interval, the computer will wait until the previous cycle is completed.

When main memory is referenced for a store operation, the computer is immediately ready for useful computing. However the full memory cycle time must elapse before another memory reference can be made. Therefore since the store operation requires a single clock pulse, the "shadow time" is 4 clock pulses, and if another main memory reference is made during this time the computer will wait until the previous cycle is completed. It is the programmer's responsibility to utilize the "shadow time" effectively if he can. Since fast memory read-out is non-destructive, no "shadow time" results from a fast memory access. The amount of time lost in storing into fast memory (no computing is possible until such an operation is completed) is 8 microseconds. This operation should probably be avoided if maximum speed is desired.

Each individual module of main memory has its own read-write circuitry, so that reference to module 2 may, if desired, be made during the "shadow time" resulting from a reference to module 1, with no penalty in execution speed.

In the micro-steps described below, the R1 field designates a register which contains an address. The low order 15 bits of this register (bit positions 9-23) are used as the address; the high-order bits are ignored. The carry toggles are not affected unless specifically stated in the description. The parity toggle is set to 1 if the

number of 1's in the word is even, or to 0 if the number of 1's is odd.

Mnemonic	Meaning	Description
LDM R1 R2	Load from Memory	The content of the memory cell, designated by the address in register R1, replaces the content of the register designated by R2.
STM R1 R2	Store into Memory	The content of the register designated by R2 is stored into the memory cell designated by the address in R1. The content of R1 and R2 remains unchanged.
LDI R1 R2	Load (from memory) and Increment	The content of the addressed memory cell replaces the content of the register designated by R2. Simultaneously, the address in register R1 is incremented by 1 in bit position 23. This latter operation does not change the setting of the carry toggles. When R1 and R2 designate the same register the content of the addressed memory cell and the incremented value of the designated register are combined (logical or), and the logical sum replaces the content of that register.
STI R1 R2	Store and Increment	The content of the register designated by R2 is stored into the addressed memory cell. Simultaneously, the address in R1 is incremented by 1 in bit position 23. This latter operation does not change the setting of the carry toggles. If R1 and R2 designate the same register, the memory cell will contain its own address and R1 will contain the logical sum of the address and its incremented value at the end of the operation.

It is clear that the above micro-steps can refer to any memory cell in the computer, but an address for that cell must be present in one of the registers. (Designating Q or N by the R1 field can refer only to main memory location 00000. This should probably be avoided.) Since many common operations involve the requirement that the content of some register be stored temporarily, a group of 8 cells in main memory has been designated as "working storage". These may be referred to directly, without the requirement that an address be provided. The following two micro-steps reference these cells. The R1 field designates which one of the 8 cells is to be used, and *does not* designate a register.

Mnemonic	Meaning	Description
LDW R1 R2	Load from Working Storage	The content of the working storage cell designated by R1 replaces the content of the register designated by R2. The content of the designated memory cell remains unchanged.
STW R1 R2	Store into Working Storage	The content of the register designated by R2 is stored into the designated working storage cell. The content of the register designated by R2 remains unchanged.

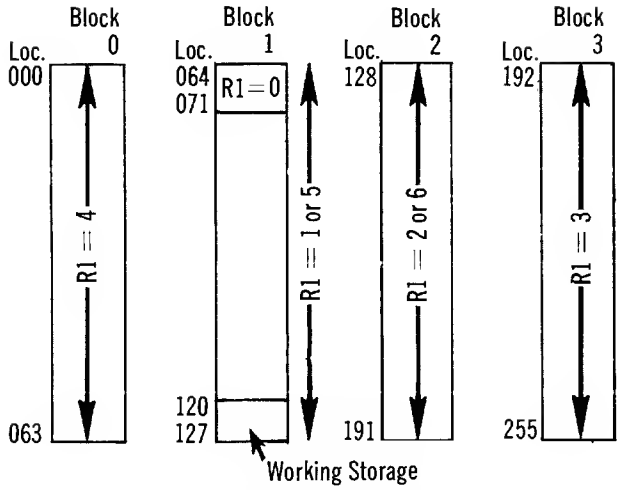
LDS and STS—Load Special and Store Special

In addition to the above, two special micro-steps have been provided which reference certain fixed locations in memory. In these micro-steps the R1 field is used as a control field to designate which one of eight possible load or store operations is desired. In all of the operations, the effective address of the cell is constructed from a designated field of the D register.

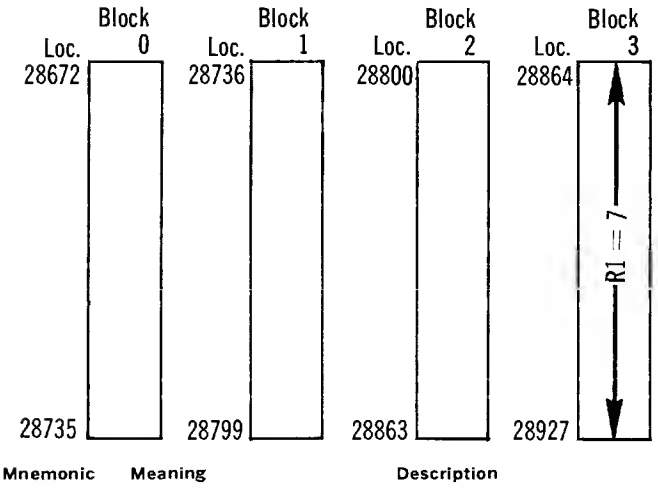
The first portion of main memory is divided into blocks of 64 words each, numbered, for reference, blocks 0-3. Seven of the 8 possible configurations of R1 reference this portion of memory while the eighth configuration references fast memory according to the following scheme:

R1	Portion of D Used	Memory Block Referenced
0	Bits 6-8 (mode-field)	Block 1, cells 00-07 (Locations 64-71)
1	Bits 18-23	Block 1, cells 00-63 (Locations 64-127)
2	Bits 18-23	Block 2, cells 00-63 (Locations 128-191)
3	Bits 18-23	Block 3, cells 00-63 (Locations 192-255)
4	Bits 0-5	Block 0, cells 00-63 (Locations 00-63)
5	Bits 0-5	Block 1, cells 00-63 (Locations 64-127)
6	Bits 0-5	Block 2, cells 00-63 (Locations 128-191)
7	Bits 0-5	Fast memory cells 28864 to 28927 (last 64 cells in 1st module of fast memory)

The portions of main memory referenced by R1 configurations 0-6 may be visualized by the following representation of memory module 1:



The portion of fast memory referenced by R1 configuration 7 may be visualized by the following representation of memory module 8:



Mnemonic	Meaning	Description
LDS R1 R2	Load Special	An address is constructed from the content of portions of the D register, as designated by R1. The content of the addressed cell replaces the content of the register designated by R2.
STS R1 R2	Store Special	An address is constructed from the content of portions of the D register, as designated by R1. The content of the register designated by R2 is stored in the addressed memory cell. The content of the register designated by R2 remains unchanged.

Several useful operations can be performed simply and easily by means of these micro-steps. For example, any arbitrary 6-bit character code can be converted into any other code by having available a table of the replacement code, arranged in the proper order, in memory. When one character of the alien code is obtained it is placed in the D register, say in bit positions 18-23. A single micro-step, e.g., "LDS 3 D" will replace the character with one chosen from the table in memory block 3. Another extremely important operation involves branching to one of 64 possible microutines on the basis of a macro-instruction operation code. If this code is located in, say, positions 0-5 of the D register, and a table of microutine starting locations is located in memory block 0, then execution of the micro-step "LDS 4 P" will cause the desired branch of control. If a faster branch is desired, in which fast memory is used for the "jump table" (located in the last 64 words of the first 256-word block of fast memory) the micro-step "LDS 7 P" will effect the control transfer.

It should be clear that memory location assignments for macro-instructions must take into account those main memory locations actually used by the instruction set, including working storage. It is expected that normal operation will involve using the first 64 words of main memory (Block 0) as a "jump table" for rapid macro-instruction interpretation, and that Block 1 will be used for index registers and, possibly, Block 1, 2, or 3 to replace the character set obtained from input devices with one more amenable to programming use.

Logical Word Operations

A rather complete set of operations is provided for the manipulation of a 24-bit word, called a logic word. In all of the following operations all 24 bit positions take part in the operation; where registers of less than 24 bits in length are involved, the "missing" portions take place as if they contained zeroes. In all of these micro-steps R1 and R2 designate registers, and in all cases the result of the operation appears in the register designated by R2. The carry toggles are not affected unless specifically stated in the description.

Mnemonic	Meaning	Description
CPL R1 R2	Copy Logical	The content of the register designated by R1 replaces the content of the register designated by R2. The content of register R1 remains unchanged. If both R1 and R2 designate the same register, no change is observed.
CCL R1 R2	Copy Complement Logical	The content of the register designated by R1 is complemented (1's complement) and this result

Mnemonic	Meaning	Description
		replaces the content of the register designated by R2. The content of register R1 remains unchanged unless it is the same register that is designated by R2.
CIL R1 R2	Copy and Increment Logical	A 1 at bit position 23 is added to the content of the register designated by R1, and this result replaces the content of the register R2. All three carry toggles are set by this operation; they will be set to 0 if no carry-out passes their bit position, and will be set to 1 if such a carry is present. Register R1 remains unchanged unless it is also designated by R2.
CDL R1 R2	Copy and Decrement Logical	An internally generated word, consisting of 24 1's, is added to the content of the register designated by R1. This result replaces the content of register R2. The net effect is the same as if a 1 at bit position 23 had been subtracted from the content of R1. The carry toggles are *not* set by this operation, and register R1 remains unchanged unless it is also designated by R2.
EXC R1 R2	Exchange	The contents of the two registers designated are exchanged. It doesn't matter which of the two is designated by R1 and which by R2; the result is the same. If R1 and R2 designate the same register, no change is observed.
AND R1 R2 or EXT R1 R2	And Extract	The logical product of the contents of the two registers replaces the content of the register designated by R2. The result has a 1 only in those bit positions where both words contained a 1. The content of R1 is unchanged.
LOR R1 R2	Logical Or	The logical sum of the contents of the two registers replaces the content of the register designated by R2. The result has a 1 in any bit position where either of the words contained a 1. The content of register R1 is unchanged.
XOR R1 R2	Exclusive Or	The logical operation "exclusive or" is performed on the two designated words, and the result replaces the content of the register designated by R2. The result has a 1 in any bit position where one word had a 1 and the other did not; in bit positions where both words contained a 1, or both contained a 0, the result is 0. Note that if R1 and R2 designate the same register, the result is an all-zero register.
ADL R1 R2	Add Logical	The arithmetic sum of the two designated registers replaces the content of the register designated by R2. All three carry toggles are set by this operation. Register R1 remains unchanged unless it is the same register designated by R2. In this case the result is identical with that obtained by shifting the content of the register left 1 position.
ALC R1 R2	Add Logical for Carry	The three carry toggles are set by this operation, based on whether or not a carry-out would occur at their bit position if the contents of the two designated registers were added together arithmetically. The contents of the registers designated by R1 and R2 remain unchanged.
SLS R1 R2 or SL6 R1 R2	Shift Left Six	The content of the register designated by R1 is shifted left 6 bit positions, end-around, and the result replaces the content of the register designated by R2. The content of register R1 remains unchanged unless it is also designated by R2.

Partial-Word Operations

Many of the micro-steps reference certain portions of the 24 bit word, and operate only on that portion. The remainder of the word remains unchanged by the operation. As with the logic word format micro-steps, R1 designates one of the addressable registers and R2 designates the other, with the result appearing in R2. The carry toggles are not affected unless the description so states.

Mnemonic	Meaning	Description
CPS R1 R2	Copy Sign	The sign position (bit 0) of the register designated by R1 re-

Mnemonic	Meaning	Description
		places the sign position of the register designated by R2. The remainder of register R2 remains unchanged. Register R1 is not affected by this operation. If R1 and R2 designate the same register no change is observed.
CCS R1 R2	Copy Complement Sign	The sign position of register R1 is copied into the sign position of R2 in complemented form. If the sign of register R1 were positive (contained a zero) then the sign position of register R2 will be negative (will contain a 1) after the operation. If R1 and R2 designate the same register, the net effect is to change the sign of that register.
ADS R1 R2	Add Signs	The sign position of register R1 is added to the sign position of register R2, the result replacing the sign of register R2. Carry toggle 0 will be set by this operation. It will be *on* (will contain a 1) if the sign of both designated registers contained a 1, and will contain a zero (be set *off*) otherwise. The sign of register R1 is unchanged unless it is also designated by R2.

These three micro-steps give the programmer complete control over the sign positions of registers "A", "B", "C", and "D", the only registers whose sign position is actually present. Reference may be made to the other registers, particularly in the R1 field of the micro-step, where this might be useful. Remembering that any "missing" portion of a register may be thought of as containing zeroes, the micro-step "CPS N A" will set the sign of the "A" register positive. Similarly, "CCS L B" will set the sign of "B" negative. The sign position of a register may be "preserved" (in carry toggle 0) by the micro-step "ADS C C". The result of this operation will leave the sign of the "C" register zero, and carry toggle 0 will be *on* only if the original content of the sign position was a 1 (was negative).

It should also be noted that "ADS A D" will leave the sign of the "D" register in a state which indicates whether it was the same as the sign of the "A" register or not. If the signs of "A" and "D" were both positive, or were both negative, the D register sign will be positive following the operation. It will be negative otherwise. This operation corresponds to the logical operation of "exclusive or", applied only to the sign positions of the two registers. The micro-step "CCS B B" will, of course, reverse the sign of the "B" register.

Mnemonic	Meaning	Description
CPM R1 R2	Copy Magnitude	The magnitude field (bits 1-23) of the register designed by R1 replaces the same field of the register designated by R2. The sign position of the R2 register remains unchanged. If both R1 and R2 designate the same register, no change is observed.
CCM R1 R2	Copy Complement Magnitude	The magnitude field of the register designated by R1 is complemented (1's complement) and this result replaces the same field of the register designated by R2.
ADM R1 R2	Add Magnitude	The magnitude field of the register designated by R1 is added to the magnitude field of the register designated by R2. This result replaces the magnitude field of the R2 register. The sign position of the R2 register is not affected. Carry toggles 1 and 9 are both set by this operation, but carry toggle 0 is not affected. If R1 and R2 designate the same register, the result will be the same as a magnitude left-shift of 1, with carry toggle 1 indicating the value of the bit "shifted off".
AMK R1 R2	Add Magnitude with Carry-in	The magnitude field of the register designated by R1 is added to the magnitude field of the register designated by R2, and, at the same time, the content of carry toggle 1 is added into bit position 23 of the sum. Carry toggles 1 and 9 are set by this operation, but carry toggle 0 is not affected.

The micro-steps described above allow the computer to perform arithmetic computations using data words arranged in a "sign-and-magnitude" format, providing the sign of the sum is determined separately and is inserted into the sign position of the proper register. Double-precision addition in this format is particularly easy providing the format does not require that the sign position of the least significant word be used. For example, if the "A" and "B" registers contain one double-

precision data word and the "C" and "D" registers contain the other, the micro-sequence

ADM D B

AMK C A

places the double-precision sum in the A and B registers. In the case of differing signs, the programmer must expand this sequence to perform the complement operation on the proper words, and to establish the sign of the result.

Certain of the micro-operations have been chosen to allow rapid manipulation of double-precision data words in a particular format; this format is shown in Figure 3 and is designated the "floating-point format". It should not be assumed that the format shown is the only possible one which can be implemented on the computer; it is one whose execution times will be particularly short. The micro-steps described below reference the "exponent" and "fraction" fields of the floating-point format. It should be noted that the "fraction" field of 15 bits is also a convenient field for an address in a macro-instruction word, and the micro-steps dealing with this field can also be used to single out the address field when it corresponds.

Mnemonic	Meaning	Description
CPX R1 R2	Copy Exponent	The exponent field of the register designated by R1 replaces the exponent field of the register designated by R2. Bit 0 and the bit field 9-23 of the R2 register are not affected by this operation. The exponent field in any register may be set to zero selectively if the R1 field references a register whose exponent field is not present (is permanently zero).
CCX R1 R2	Copy Complement Exponent	The exponent field of the register designated by R1 is complemented (1's complement) and the result replaces the exponent field of the register designated by R2. If R1 references a register whose exponent field is permanently zero, and R2 designates a full 24-bit register, the exponent field in the latter register is set to all 1's.
CIX R1 R2	Copy and Increment Exponent	A 1 at bit position 8 is added to the exponent field of the register designated by R1, and the result replaces the exponent field of the register designated by R2. Carry toggle 1 is set by this operation, and will contain a 1 only if the original exponent field of the R1 register contained all 1's. Carry toggles 0 and 9 are not affected by this operation.
ADX R1 R2	Add Exponents	The exponent field of the register designated by R1 is added to the exponent field of the register designated by R2, and the sum replaces the exponent field of the R2 register. Carry toggle 1 is set by this operation; the other carry toggles are not affected.
CPF R1 R2	Copy Fraction	The fraction field (bits 9-23) of the register designated by R1 replaces the fraction field of the register designated by R2. Bits 0-8 of the R2 register are not affected by this operation. The fraction field in any register may be set to zero if the R1 field references a register whose fraction field is not present.
CCF R1 R2	Copy Complement Fraction	The fraction field of the register designated by R1 is complemented (1's complement) and the result replaces the fraction field of the register designated by R2. If R1 references a register whose fraction field is permanently zero, the fraction field in the register referenced by R2 is set to all 1's.
ADF R1 R2	Add Fraction	The fraction field (bits 9-23) of the register designated by R1 is added to the fraction field of the register designated by R2, and the result replaces the fraction field of the R2 register. Carry toggle 9 is set by this operation; the other carry toggles are not affected.
AFK R1 R2	Add Fraction with Carry-in	The fraction field of the register designated by R1 is added to the fraction field of the register designated by R2, and, at the same time, the content of carry toggle 1 is added into bit position 23 of the sum. Carry toggle 9 is set by this operation; the other carry toggles remain unchanged.

Mnemonic	Meaning	Description
		The resulting sum replaces the fraction field of the R2 register; bit positions 0-8 of the R2 register remain unchanged.

An additional micro-step, which might be included in this group, is the one designated "SFR—Shift Floating Right". Since it is somewhat specialized, and involves the use of the N register for repeat-counting, its description will be deferred until the discussion of the other shift operations.

CLN and CLD—Copy Literal to N and Copy Literal to D

Several "copy" operations of a specialized nature are possible utilizing micro-steps of this type. Some of these are described under "Transfer of Control", since this is the effect obtained. Another two micro-steps provide the same function, but reference different registers.

The term "Literal" is used in these micro-steps to designate that the operand is contained in the modifier field of the micro-step.

Mnemonic	Meaning	Description
CLN M	Copy Literal to N	The Modifier field of this micro-step, treated as a single 6-bit unit, is copied into bit positions 3-8 of the N register. Bit positions 1 and 2 of the N register are set to zero by this operation. This micro-step provides a simple way of initializing the "N" register for repeated operations such as multiplication, division, and the various shifting micro-steps.
or		
LRC M	Load Repeat Counter	
CLD M	Copy Literal to D	The modifier field of this micro-step, treated as a single 6-bit unit, is copied into bit positions 18-23 of the "D" register. Bit positions 0-17 are set to zero by this operation.

CFS and CTS—Copy from Special and Copy to Special

This pair of micro-steps is provided primarily for the purpose of program interrupt operation. They allow the programmer to record into the designated register the contents of the various toggles for storage into main memory, and provide the facility of retrieving this information after the interrupt operation is complete and restoring the original state of the toggles. Included as well is the facility for setting special (optional) registers connected with high-speed input-output devices. It is also possible to set an information pattern into any class of toggle, or into all three classes together.

In these micro-steps, the R1 field designates the class, or classes, of device which is to be referenced, according to the following table:

R1	Items Referenced
4	All Carry Toggles
5	All Interrupt Masks
6	All Program Flags
7	All Toggles (Items 4, 5, and 6 Above)

The R2 field is used to designate a register which either receives the information from the various items, or provides the information for setting the items referenced by the R1 field. By convention, the various classes of toggles are copied to or from the following register bit positions:

1. Carry toggles 0, 1, and 9 correspond to register bit positions 9-11.
2. The interrupt masks correspond to register bit positions 14-17.
3. The program flags correspond to register bit positions 18-23.

Mnemonic	Meaning	Description
CFS R1 R2	Copy from Special	The information contained in the item, or items, designated by R1 is copied into the register designated by R2. If the information does not occupy the complete register, the "unused" portions of the register are set to zero. Where toggles are designated by R1, if the toggle is in the *on*, or *true*, state, the register bit position corresponding will be set to 1. If the toggle is off (false state) the bit position will be set to zero.
CTS R1 R2	Copy to Special	The information contained in the register designated by R2 is copied into the item, or items, designated by R1, where such an operation is possible.

Shifting Operations

All shifting operations are performed in a repeated fashion (with the exception of the "SLS—Shift Left 6" micro-step) and utilize the "N" register for counting purposes. Execution proceeds at a rate of 1 clock pulse per bit shifted (a shift of either zero or 1 position requires 1 clock pulse). For the direct-shift micro-steps, the programmer must initialize the "N" register prior to

execution of the shift micro-step. A count may be copied into the "N" register from some other register; it may be loaded from memory, or may be inserted by means of the "LRC—Load Repeat Count" micro-step. Execution of the direct-shift micro-steps will repeat until the "N" register is "counted down" to zero. The carry toggles are not affected by these micro-steps.

SSL and SDL—Shift Single Length and Shift Double Length

These two micro-steps utilize the R1 field to designate the type of shifting operation desired. To simplify discussion, the three bit positions of the R1 field are called bits A, B, and C. In these two micro-steps they are given the following meanings:

Bit	Value	Description
A	0	Shift Left
A	1	Shift Right
B	0	Closed (End-Around) Shift
B	1	Open Shift
C	0	Magnitude Shift (sign position does not take part in the operation)
C	1	Logical Shift (sign position is shifted)

When an "open" shift is indicated, the vacated positions of the register are set to zero, and the bits "shifted off" are lost. The "N" register will always contain zeroes after the execution of a direct-shift operation.

Mnemonic	Meaning	Description
SSL R1 R2	Shift Single Length	The content of the "N" register is examined. If it is initially zero, no operation takes place. If it is non-zero, the content of the register designated by R2 is shifted one bit position in the manner specified by R1, and the content of the "N" register is decremented by 1. This process is repeated until the "N" register is reduced to zero.
SDL R1 R2	Shift Double Length	The coupled contents of the "A" and "B" registers, treated as a single double-length register, are shifted in the manner designated by R1. The R2 field of this micro-step is not used. The "N" register is decremented by 1 for each bit position shifted. The operation is repeated until the "N" register is reduced to zero.

SFR—Shift Floating Right

This micro-step is provided to facilitate certain floating-point operations:

Mnemonic	Meaning	Description
SFR R1 R2	Shift Floating Right	The contents of the fraction field of the "A" register and the magnitude field of the "B" register are treated as a single, extended-length register and are shifted right until the content of the "N" register is reduced to zero. Zeroes fill the vacated portion of the "A" register, and bits shifted beyond position 23 of the "B" register are lost. The R1 and R2 fields are not used. The sign and exponent fields of the "A" register and the sign position of the "B" register are unaffected.

SLC—Shift Left and Count

Although designed primarily as an aid to manipulations involving normalized floating-point numbers, this micro-step can be utilized for various types of logical operations as well. The magnitude field of the "B" register, and designated portions of the "A" register, are treated as a single extended-length register for this operation. The R1 field of the micro-step serves two functions: It designates which portions of the "A" register are to be involved in the shifting operation, and also indicates which bit position, or positions, are to be examined for termination of the shifting process. The R2 field is not used.

The shifting operation is always a left shift, and is repeated until one of the bit positions designated for examination contains a 1. There are circumstances in which this operation will not terminate, and it is the programmer's responsibility to see that they do not occur. For each bit position shifted the "N" register is decremented by 1, but the "N" register is not examined by the process; it terminates only when one of the designated bit positions of the "A" register is non-zero.

The effect obtained from the various possible configurations of the R1 field can best be seen by reference to a table. The letters "S", "X", and "F" are used to designate the sign field, the exponent field, and the fraction field respectively. Where more than one letter appears it indicates that the designated fields take part in the shift; absence of a letter or letters indicates that the fields do not take part in the operation. The R1 field is shown in binary format.

R1	Portion of "A" Shifted	Bit Positions of "A" Examined
000	F	None. This will not terminate
001	F	Position 9 only
010	XF	Position 1 only
011	XF	Positions 1 and 9
100	SXF	Position 0 only
101	SXF	Positions 0 and 9
110	SXF	Positions 0 and 1
111	SXF	Positions 0, 1 and 9

Mnemonic	Meaning	Description
SLC R1 R2	Shift Left and Count	The coupled contents of the magnitude field of the "B" register and designated portions of the "A" register are treated as a single extended-length register and are shifted left until one (or more) of the bit positions of "A" designated for examination contains a 1. The content of "N" is decremented by 1 for each bit position shifted. The sign position of "B" and portions of "A" not designated for shifting remain unchanged. Vacated portions of "B" are set to zero.
or		
FLC R1 R2	Float Left and Count	

Multiplication and Division

Two micro-step operations are provided which can greatly simplify the arithmetic operations of multiplication and division; in many cases they are sufficient in themselves to provide this function. They are both repeated micro-steps, and the programmer must specify, by establishing the proper count in the "N" register, how many times they are to be repeated. The multiplication operation (MPS—Multiply Step) requires 1 clock time per step of its execution, while divide step (DVS) requires 2. The operation of each is perhaps best explained by means of a flow chart (Figure 4). The conditions which must be established by the programmer prior to the execution of MPS are as follows:

1. The multiplier is loaded into the "B" register.
2. The multiplicand is loaded into the register designated by R1 (normally the "C" or "D" register).
3. The repeat count is loaded into the "N" register.
4. If the result is to be a simple double-length product the "A" register must be set to zero. If it is not zero, its content will be added into the least significant bit positions of the double-length product.

Assuming the "N" register contains "23" at the start of the operation, indicating that the multiply step operation is to be executed 23 times, the results will be as follows:

1. The "A" register will contain the most significant bits of the product in bit positions 1-23. Bit position 0 of the "A" register does not take part in the operation and remains unaffected.
2. The "B" register will contain the least significant half of the double-length product in bit positions 1-23. Bit position 0 of the "B" register does not take part in the operation and is unaffected.
3. The "N" register will contain all zeroes.
4. The contents of all the other registers (except, perhaps, "P") remain unaltered.

This multiplication algorithm has many advantages which may not be obvious from casual inspection. First of all, the sign of the product can be established in the single micro-step "ADS". For integer multiplication (where the least significant bit is located in bit position 23) the multiplicand is loaded into the "B" register, "A" is set to zero (if no addition is to take place) and the "MPS" micro-step is executed 23 times. The integer product will appear in the "B" register, and overflow will be indicated by a non-zero "A" register. In this case, the sign of the product would be established in the "B" register, either before or after the operation.

Again, dealing with integers, a repeated multiply step operation can give the result B = BC + A, where the letters designate the registers involved, and the "=" sign indicates that the result appears in "B". As before, a non-zero "A" register indicates overflow.

The divide-step algorithm, shown in flow chart form in Figure 4, has similar advantages. Prior to execution of the DVS operation the programmer must establish the following conditions:

1. The most significant half of the dividend must be in the "A" register in bit positions 1-23. The sign bit of "A" is unimportant, but will be destroyed in the process.
2. The least significant half of the double-length dividend must be in the "B" register, bit positions 1-23. The sign position of the "B" register does not take part in the operation, and may contain the pre-established sign of the quotient if desired.
3. The divisor, in 2's complement form, must be in bit positions 1-23 of the "C" or "D" register. The sign position of that register must contain a 1.
4. The repeat count must be in the "N" register.

Assuming the "N" register contains "23" at the start of the operation, indicating that the divide step opera-

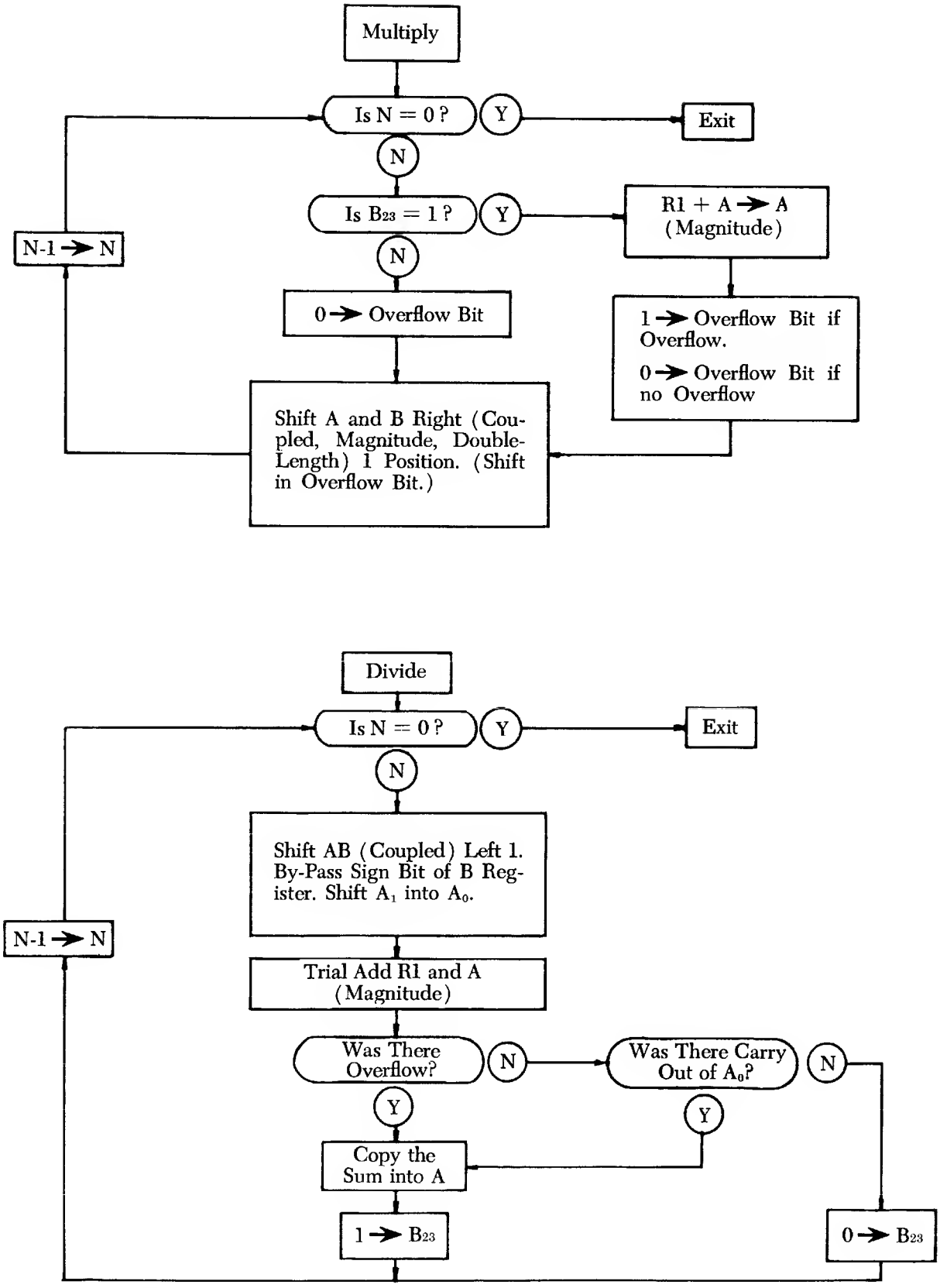


Figure 4. Multiply and Divide Operations

tion is to be executed 23 times, the results will be as follows:

1. The quotient will appear in the "B" register in bits 1-23.
2. The remainder will appear in the "A" register, in bit positions 1-23. The sign of the "A" register will, in general, not mean anything; this can best be seen from the flow chart.
3. The "N" register will contain all zeroes.
4. The other registers (except, perhaps, "P") will be unchanged.

No automatic provision is made to detect a divide-overflow condition. This problem is left as an exercise for the programmer.

While it is, in theory, possible to designate some register other than "C" or "D" in using both MPS and DVS, this should be done with caution; in particular, designating "N" as the multiplicand will result in the multiplicand being decremented by one after each iteration. The two micro-steps use the same format as the others, except that the R2 field *must* designate the "A" register. Neither micro-step affects the carry toggles.

Mnemonic	Meaning	Description
MPS R1 A	Multiply Step	The content of the register designated by R1 is multiplied by the content of the "B" register, and the double-length product appears in the magnitude fields of "A" and "B". The "N" register must initially contain the number of multiplication iterations desired. The sign positions of the "A" and "B" registers are not affected. The prior content of the "A" register is added into the least significant bit positions of the double-length product. The "N" register will contain all zeroes.
DVS R1 A	Divide Step	The content of the coupled "A" and "B" registers, treated as containing (in their magnitude fields) a double-length dividend, is divided by the content of the register designated by R1. This latter register must contain the divisor in 2's complement form, and the "N" register must initially contain the number of divide iterations desired. Following execution, the quotient replaces the magnitude field of the "B" register, and the remainder replaces the magnitude field of the "A" register. The sign of the "B" register is unaffected, but the sign of the "A" register is destroyed. The "N" register will contain all zeroes.

Transfer of Control

The "P" register contains the address in which the next micro-pair to be executed is located. If the content of the "P" register is altered during the execution of any micro-pair, the new pair will be obtained from the (altered) location indicated by the "P" register. Three special micro-steps are provided to facilitate this process, although it may also be accomplished by other means. For example, if a transfer to some specified location is desired, it can be accomplished by the micro-step "LDM P P". In this case the transfer address must be located in the memory cell following the one in which the LDM micro-step is located. Transfer to any portion of memory can be effected in this manner. If return-transfer is desired (that is, if one wants to return to this sequence after transfer to another, as in a subroutine call) the micro-pair

```
LDI  P D
EXC  P D
```

will cause transfer to the desired location in memory, with the proper return location in the "D" register. In general, any alteration in the "P" register content will cause a transfer of control unconditionally. One interesting point in this regard is the following: Recalling that the "N" register and the "P" register do not overlap, the micro-step

```
AMK  N P
```

will cause the following micro-pair to be skipped (not executed) if carry toggle 1 contained a 1; the pair will be executed if it contained a zero.

The following three micro-steps utilize the modifier field as a unit (see Figure 3); it is treated as a single 6-bit (literal) field. Carry toggles are not affected by these micro-steps.

Mnemonic	Meaning	Description
CLP M	Copy Literal to P	The content of the M field replaces bit positions 18-23 of the "P" register. The rest of the "P" register is set to 0 except for bit positions 9-11, which are set to 1. This results in an address which refers to one of the first 64 words of fast memory.
ALP M	Add Literal to P	The content of the M field (from this micro-step) is added

Mnemonic	Meaning	Description
or		into bit positions 18-23 of the "P"
FTR M	Forward Transfer Relative	register. The new micro-pair to be executed will be obtained from the resulting address. Remember that "P" contains, normally, the address of the *next* micro-pair to be executed.
ACP M	Add Complement (of Literal) to P	A number is added to the content of the "P" register consisting of all 1's in bit positions 9-17, and the literal field M in positions 18-23. The M field should be in 1's complement form to effect a transfer of control to an earlier part of the sequence. Remember that "P" normally contains the address of the *next* micro-pair to be executed; if the M field is composed of all 1's (the complement of zero) a dynamic halt will result.
or		
BTR M	Backward Transfer Relative	

The last two micro-steps provide simple means for a relative transfer of control for looping and for other operations, while the first of the trio provides a method of control transfer to a group of fixed locations in fast memory. Since each command set requires a control sequence, this could be located in this region for simple return-transfer from the many different microroutines.

Test Operations

Several micro-steps are provided for testing bits within registers and most of the hardware toggles. In all instances, the address field of the micro-step specifies the item (bit or toggle) being tested. The response is a conditional prompt "skip" of the next micro-step or micro-pair. If the conditional-test micro-step is the left-half micro-step, the right-half micro-step will be conditionally skipped; if it is in the right-half, the next complete micro-pair will be conditionally skipped.

When a left-half test micro-step is executed, its execution time, as usual, requires one clock pulse. If the condition is met (i.e., the test is "successful") the right-half micro-step will be executed as written and its timing will also be normal. If, however, the condition was not met, the right-half micro-step is not executed, and the next micro-pair is inserted into the "E" register during the next clock pulse.

Execution of a right-half test micro-step precludes obtaining the next micro-pair until the result of the test is known. Thus, if the test is successful, the total elapsed time is two clock pulses. If the test is unsuccessful, the computer increments "P" and places the word addressed by "P" into the "E" register during the same clock pulse. Therefore, a total elapsed time of two clock pulses is required for the execution of a right-half test micro-step, whether the condition is successful or not.

TZO and TNZ—Test for zero and test for non-zero

These two micro-steps can address any one of the eight working registers shown in Figure 2. The R1 field of the micro-step is used to designate the register, while the R2 field is used to designate the specific portion, or portions, of the register to be tested. The high order bit of the R2 field designates the sign position of the register, the next (or center) bit designates the exponent field (bit positions 1-8) and the least significant bit designates the fraction field. If it is desired to test a portion of a register, the corresponding bit must be a 1. If a bit in R2 is a 0, the corresponding portion of the register will not be tested.

For sign-testing, only the sign position should be designated. The convention adopted in the design specifies that a zero sign position is used to designate a plus (or false) state, while a 1 is used to designate a minus (or true) state. The bit pattern of the R2 field may be used in any combination desired; an octal 7 (all three bit positions 1) will test the entire 24 bit register; an octal zero will cause no skip, i.e., the condition tested is considered met.

Mnemonic	Meaning	Description
TZO R1 R2	Test for Zero	The register designated by R1 is tested for zero in those fields specified by R2. If the indicated register contains only zeroes in the field(s) specified for testing, the next sequential micro-step will be executed. If the register is not all zero in the field(s) specified for testing the next micro-step (or micro-pair) will be "skipped".
TNZ R1 R2	Test for Non-Zero	The register designated by R1 is tested for zero in those fields specified by R2. If the indicated register contains only zeroes in the field(s) specified for testing, the next micro-step (or micro-pair) will *not* be executed. If the register is not all zero (is non-zero) in the field(s) specified for testing, the next micro-step (or micro-pair) *will* be executed, since the specified condition (non-zero) is met.

TMT and TMF—Test Mode bits for true and Test Mode bits for false

This pair of conditional-test micro-steps is provided to give extensive bit-testing operations on one specified field of the "D" register. This field (called the mode field, or mode bits) can be tested for any desired bit pattern. It is located in bit positions 6-8 of the "D" register.

It is expected that most instruction sets designed for the PB440 will utilize this field to specify index registers or special addressing modes, and this pair of micro-steps is provided to make interpretation and index-register reference as simple and direct as possible.

In both of these micro-steps, the bit pattern to be tested for appears as the R1 field. "Ones" in the R2 field are used to indicate which of the three bit positions in the "D" register are to be tested. If R2 contains an octal 7, for example, all three of the mode bits of the "D" register will be tested. If it contains a zero, none of the mode bits will be tested, and the next micro-step (or micro-pair) will be executed.

If the bit pattern which comprises the R1 field is identical, in those bit positions specified for testing by the mask in R2, the condition is met; if it is not identical, the condition is not met.

Mnemonic	Meaning	Description
TMT R1 R2	Test Mode Bits (of "D") for True	The mode bits of the "D" register (bit positions 6-8) are tested against the bit pattern shown in R1, in those positions designated by a mask in R2. If the pattern is identical, (the condition is met) the next micro-step (or micro-pair) is executed. If the tested condition is not met (the pattern is not identical) the next micro-step (or micro-pair) is skipped.
TMF R1 R2	Test Mode Bits (of "D") for False	The mode bits of the "D" register (bit positions 6-8) are tested against the bit pattern shown in R1, in those bit positions designated by a mask in R2. If the pattern is identical (the condition is met) the next micro-step (or micro-pair) is *not* executed. If the tested condition is not met (the pattern is not identical) the next micro-step (or micro-pair) is executed.

TCT and TCF—Test Condition True and Test Condition False

This pair of micro-steps is provided to test various conditions not covered by the zero test and mode test micro-steps. In these micro-steps, the R1 field is used to designate which *class* of device is to be tested, and R2 designates which member (or members) of that class is to be tested. The device classes which can be tested in this way are:

Class 0:

The six program flags. These are individual toggles which can be set under program control to either the true or false state; they may be considered to be 1 bit (Boolean) registers.

Class 1:

Interrupt masks. These masks (toggles) are used to control the effect of interrupt signals received from the input-output devices. They are described in more detail in the section concerned with input-output equipment.

Class 2:

The carry toggles and the parity toggle. A parity toggle test should be made with caution since the results being tested may be destroyed by an interrupt before the test is made. This possibility may be voided by making certain the micro-step making the test is the right-hand micro-step of the word containing the load or store micro-step which sets the parity toggle for the test. It may also be avoided by turning *off* the interrupt masks for all channels that could cause an interrupt until the test has been made.

Class 3:

The sense switches. These switches are mounted on the computer console and can be set by the operator. Their setting cannot be altered by a program in the computer.

Class 4:

Interrupt lines. These lines are made *true* or *false* by the input-output devices, and represent one way of determining the status of any device. Their operation is described in more detail in the section dealing with the input-output equipment.

Class 5:

"A" or "B" register bit positions. Certain of these bit positions can be tested individually for 0 or 1.

Class 6:

"D" register bit positions. Certain specific bit positions of the "D" register may be tested individually for 0 or 1.

Class 7:

Not assigned.

Some possible combinations represented by R1 and R2 have not been defined. Those which have been defined are shown in the following table:

R1 (Class)	R2	Item
0	1	Program Flag #1
0	2	Program Flag #2
0	3	Program Flag #3
0	4	Program Flag #4
0	5	Program Flag #5
0	6	Program Flag #6

R1 (Class)	R2	Item
1	1	Interrupt Mask for Channel #1
1	2	Interrupt Mask for Channel #2
1	3	Interrupt Mask for Channel #3
1	4	Interrupt Mask for Channel #4
2	0	All Carry Toggles Zero (Off)
2	1	Carry Toggle 0
2	2	Carry Toggle I
2	3	Carry Toggle 9
2	7	Parity Toggle (This toggle indicates the parity of the last word involved in a memory access micro-step.)
3	0	All Sense Switches Off
3	1	Sense Switch #1
3	2	Sense Switch #2
3	3	Sense Switch #3
3	4	Sense Switch #4
3	5	Sense Switch #5
3	6	Sense Switch #6
4	0	All Interrupt Lines False. (No interrupt present)
4	1	Interrupt Line #1
4	2	Interrupt Line #2
4	3	Interrupt Line #3
4	4	Interrupt Line #4
4	5	Buffer Channel Request Line
5	1	"A" Register, Bit Position 1
5	2	"A" Register, Bit Position 23
5	7	"B" Register, Bit Position 23
6	1	"D" Register, Bit Position 9
6	2	"D" Register, Bit Position 10
6	3	"D" Register, Bit Position 11

Mnemonic	Meaning	Description
TCT R1 R2	Test Condition True	The condition designated by the R1 (class) and R2 (item) fields is tested. If the condition is *true,* the next micro-step (or micro-pair) is executed. If the condition tested is *false,* the next micro-step (or micro-pair) is skipped.
TCF R1 R2	Test Condition False	The condition designated by the RI and R2 fields is tested. If the condition is *false,* the next micro-step (or micro-pair) is executed. If the condition is *true,* the next micro-step (or micro-pair) is skipped.

STT and STF—Set Toggle True and Set Toggle False

These two micro-steps have been provided to allow the programmer to set the program flags, interrupt masks and carry toggles either "on" (true) or "off" (false). As with the "test condition" micro-steps, the R1 field is used to designate the class of device, and the R2 field is used to designate which item (or items) in the class is to be affected. It will be noted that the same class designations apply to both sets of micro-steps, except those which cannot be changed by the computer program are not defined for the "set" micro-steps. Those combinations which can be used are shown in the following table:

R1 (Class)	R2	Item(s)
0	1	Program Flag #1
0	2	Program Flag #2
0	3	Program Flag #3
0	4	Program Flag #4
0	5	Program Flag #5
0	6	Program Flag #6
0	7	All Program Flags
1	1	Interrupt Mask for Channel #1
1	2	Interrupt Mask for Channel #2
1	3	Interrupt Mask for Channel #3
1	4	Interrupt Mask for Channel #4
2	1	Carry Toggle 0
2	2	Carry Toggle 1
2	3	Carry Toggle 9

Other Micro-Steps

Mnemonic	Meaning	Description
HLT R1 R2	Halt	Execution of this micro-step causes the automatic program sequencing to stop, but has no effect on any of the registers or toggles. The R1 and R2 fields are not used.
NOP R1 R2	No Operation	Execution of this micro-step causes no operation to be performed other than (perhaps) the automatic incrementation of the "P" register. The R1 and R2 fields are not used. It does nothing at all, and it takes only 1 microsecond not to do it.
EMP R2	Execute Micro-Pair	The R2 field designates a register containing a micro-pair to be executed immediately. The content of the "P" register is not automatically incremented during execution of the special pair. If EMP is a left-hand micro-step, the right-hand micro-step will be ignored, and will require no execution time.

INPUT-OUTPUT SYSTEM

This discussion is confined to the input and output hardware of the PB440 main frame, and to micro-steps which perform input and output operations. Information on the detailed operation and control of data input and output devices is contained in a peripheral equipment manual.

An input-output device can communicate with the computer over one of four separate channels. The four channels can be assigned at will to any of the devices present on the computer, and this assignment can be changed as desired, under program control.

Each device contains a "controller" which allows the computer to exercise control over the device, and which informs the computer of the status of the device. Most controllers contain either a character or word buffer and data transmission is either in character- or word-parallel mode. The computer actually communicates only with this buffer for data transfer operations.

All data transfer and I/0 deviee control operations are under control of the computer program, and each action taken is specified by the program. Only one action at a time can be designated, but, by careful programming, many devices can be kept operating at full speed "simultaneously" since the computer program can proceed at many times the response rate of most input-output devices.

Each of the four communication channels contains an "interrupt line" which is capable of interrupting the normal computer program sequence if the programmer has made provision for this type of operation. The four interrupt lines are examined, in numerical sequence, by a four-position switch called a "commutator", at the clock rate of the computer. Each line is, therefore, examined once every four clock pulses for the presence of an interrupt signal. The programmer can control the effect that an interrupt signal can have on each of the four lines individually. The commutator action itself is also under program control. It can be "unlocked", so that it will perform its normal scanning function automatically, or it can be "locked" onto any one of the four channels. It must, in fact, be locked for any data transfer operation to take place.

Any of the devices may be "connected" to, or "disconnected" from the computer under program control, although it is unlikely that the computer will be able to turn them on or off physically. For more advanced configurations of input-output equipment, this connect-disconnect feature makes it possible for more than 4 devices to be operated on-line simultaneously.

The Interrupt Masks are a set of toggles which can be set and tested under program control. Their function is to determine the effect that an interrupt signal is to have on their particular channel. If they are set to the false or "0" state, interrupt signals have no effect on the normal program sequence. The particular interrupt line which has been "masked off" by this action will, however, remain in the *true* state, and may be tested by means of the TCT and TCF micro-steps. This mode of operation has been named "programmed interrupt", since the conditions under which the program responds to an interrupt signal are determined solely by how the program is written.

If an interrupt mask is set to the *true,* or "1", state, an interrupt signal on that channel will cause an "automatic interrupt" operation to take place. This operation consists of locking the commutator on the channel which caused the interrupt, waiting until the micro-pair in the "E" register has been executed, and then loading the "E" register from one of the first 4 locations in fast memory. If this mode of computer operation is chosen, the programmer must see to it that the proper interrupt program sequence is present in memory, and that the corresponding location chosen for its first micro-pair contains the proper micro-pair to preserve the content of the "P" register and to transfer control to the interrupt program sequence.

Since each of the interrupt masks can be set as desired, the programmer may choose between these different methods of operation for each separate channel. Any interrupt signal which reaches the commutator (that is, is not "masked off") will cause automatic interrupt and will also lock the commutator to the line on which the interrupt signal was detected. No automatic interrupts can be effected while the commutator remains locked.

An additional line (called the "Buffer Channel Request" line) is available to provide an input-output "ready" signal from remote I/0 devices. Several remote devices may be connected to this request line when the servicing of these devices is an exceptional occurrence. Devices on the request line need not be assigned an I/0 channel, and may not cause an "automatic interrupt" as described above. Instead, the devices connected to this line will provide the computer with a "ready" signal, which will be reflected in the master interrupt line (as well as the "Buffer Channel Request" interrupt line) being set "true". A micro-sequence may then interrogate the devices connected to the request line to determine which device presented the ready signal, assign an I/0 channel to the device, and then service that device. After the device is serviced, the I/0 channel may be reassigned to the device formerly on that channel.

SEL—Select an Input or Output Device

The first of the three input-output micro-steps selects a particular device, assigns it to one of the four channels, and receives a report of its status. It uses the R1 field to designate a register which must contain a "command word" for the device in question. The approximate format for this command word is shown below:

Bit Pos.	Field	Description
0	S	Status Only Bit
1-4	CO	Controller Type (Part of Device Address)
5-6	CN	Controller Number (Part of Device Address)
7-8	FM	Format Designation
9-11	DN	Device Number (Part of Device Address)
12-13	CH	Channel Assignment
14-23	OP	Operation to be Selected

The "S" field is used to select the device addressed, or to merely return a status response from that device. It acts as a master control over the bit positions in the operation field. If the S field contains a "0", the addressed device is selected and assigned to the channel number indicated in the channel assignment field. If the S field contains a "1" the operation field has no effect; it is used to obtain the status of a device without actually selecting it.

Controller Type Field

The device address actually consists of three parts: The type of controller (i.e., photoreader, typewriter, magnetic tape, etc.), the number assigned to that controller (there may be more than one of that type), and the number of the device in question, since certain controllers can handle more than a single device. The controller types already defined are as follows:

Bit Pos. 1	2	3	4	Controller
0	1	0	0	Paper Tape Punch
0	0	1	0	Paper Tape Reader
0	0	0	1	Typewriter
1	1	0	0	Magnetic Tape
1	0	1	0	Card Punch
1	0	0	1	Line Printer
1	0	1	1	Card Reader

Controller Number Field

The controller number field is defined by the particular controller addressed. Each type of controller for basic equipment has the number 00. Other controllers of that type are assigned numbers other than 00 at the time the PB440 is installed.

Format Designation Field

The interpretation of the remainder of the SEL command word (bits 9-23) is determined by the bit configuration in the "FM" field. The special interpretations of bits 9-23 are for the optional high-speed buffered I/O system.

Bits In Positions 7-8	Interpretation
00	Normal interpretation as shown above.
01	Bits 9-23 indicate the number of words of data that are to be transmitted via the optional high-speed buffered I/O system.
10	Bits 9-23 indicate the initial memory location to be associated with data transmitted by the optional high-speed buffered I/O system.
11	Bits 9-23 indicate the number of words of data that are to be transmitted via the optional high-speed buffered I/O system (An interrupt is sent upon completion of data transmission.)

Device Number Field

The device number field is defined by the particular device addressed. Each of the basic devices is assigned the number 000. Other devices of each type are assigned numbers other than 000 at the time the PB440 is installed.

Channel Assignment Field

The channel to which the device is to be assigned must appear in the "CH" field of the command word. The channel numbers are as follows:

Bits	Channel
00	Channel #1
01	Channel #2
10	Channel #3
11	Channel #4

Any device may be assigned to any of the four available channels, but should not be assigned to a channel already in use. Should two devices be assigned to the same channel and both be capable of transmitting data at the same time, a data transfer at the same time will result in both data words being combined (logical or) during transmission.

Operation Field

The operation (OP) field utilizes individual bit positions to designate the desired operation(s). More than one operation may be selected if the bits have meaning for the selected device. Certain of the operations have no meaning for certain devices, and are ignored by that controller. (You can't rewind the typewriter, for example.)

The status-response word will be returned by the controller to the register designated by R1. The status-response word is primarily to provide the computer with the status information necessary to control the input-output device. For the basic equipment this information is minimal. For optional devices this information may be quite comprehensive.

For all controllers, bit 23 of the response-word will contain a "1" if the device is present, the power is on, and the device is in working condition; otherwise that bit will contain a zero.

Mnemonic	Meaning	
SEL R1 R2	Select	The R2 field is used to designate a register which contains an input-output device command word. This command word is executed (if it indicates it should be) and the device is assigned a channel number and is "connected" to the computer on that channel. A status response word replaces the content of the register designated by R1, indicating the status of the selected device. The timing is such that R1 and R2 may designate the same register.

COM—Commutator Control

This micro-step utilizes the R1 field to indicate which of three possible operations is to be performed. Any combination of these operations is possible, and each is called into action by the presence of a "1" in the proper bit position. If we designate the three bits of the R1 field by the letters A, B, and C, the individual operations possible are as follows:

Bit	Value	Resulting Operation
A	0	Unlock the commutator for automatic scanning. If the commutator is already unlocked, no operation results.
A	1	Lock the commutator to a particular line. Note that line number one (channel #1) has the number 00, line number two has the number 01, etc. The specific line is chosen from the register designated by R2 if bit B = 1, or is its present step in sequence if bit B = 0.
B	0	No effect
B	1	Set the commutator to a value determined by the content of bit positions 21 and 22 of the register designated by R2.
C	0	No effect
C	1	The present setting of the commutator is logically "or"ed into bit positions 21 and 22 of the register designated by R2, leaving the remainder of the bit positions of that register unchanged.

Mnemonic	Meaning	Description
COM R1 R2	Commutator Control	Perform the action, or actions, indicated by the R1 field on the commutator. Utilize bit positions 21-22 of the register designated by R2 for these actions if required.

DTR—Data Transfer

The actual data transfer operation into and out of the computer is controlled by the third, and last micro-step of this group. For it to operate correctly the device must first be selected (by the SEL micro-step) and the status of the device must indicate that it is not busy. A chosen device may remain "connected" to the computer indefinitely; it is disconnected only when specifically instructed by a subsequent micro-step.

The commutator must be locked (either as caused by an automatic interrupt or by means of the COM micro-step) to the proper channel. If it is not locked, no data transfer operation will take place. The execution of the data transfer also results in a conditional jump of the next micro-step (or micro-pair) if a "special condition" line in the I/O channel is false; it will not be skipped if the "special condition" line is true. For example, if there is a parity error in the character being transmitted by the photoreader, the special condition line is true and the next micro-step (or micro-pair) is executed. If there is no parity error, the next micro-step (or micro-pair) is skipped. Thus the DTR micro- step results in a conditional jump identical to that of the "TCT--Test Condition True" micro-step, where the condition tested is the special condition line of the input-output system.

The data transfer micro-step utilizes the R1 field to indicate which of several operations is to be performed, as indicated in the following table:

R1	Operation to be Performed
0	Disconnect device.
1	Data Transfer In.
2	Data Transfer Out.
3	Data Transfer IN and OUT. This operation is provided to permit controllers which have been designed for special purposes to be used with the PB440 system.

R1	Operation to be Performed
4	No Data Transfer. The computer does not send or accept information. The controller, however, is activated to send or accept information. This combination is not practical and should be avoided.
5	Data Transfer In, and end of message.
6	Data Transfer Out and end of message.
7	Return status response from device to register designated by R2,

Mnemonic	Meaning	Description
DTR R1 R2	Data Transfer	Data are transferred into or out of the register designated by R2, under control of R1. For data transfers of less than a word of data, the character is placed in or taken from the least significant bit positions of the register. The content of the register is unchanged by the output operation. For both input and output data transfers the next micro-step (or micro-pair) is skipped if the special situation is false. If the special situation is true no skip occurs.

OPTIONAL HIGH-SPEED BUFFERED INPUT-OUTPUT SYSTEM

The PB440 standard memory module (4096 words of 25 bits including parity) is designed to be connected in a tandem arrangement to the memory busses of a single computer. Certain classes of applications, however, require that one or more modules which comprise the memory system of the computer, be shared with other devices. These applications can be accommodated by employing a memory interchange to sample read-write requests from the several devices and, to connect the device's address and data busses to those of the memory module(s) being shared for the duration of a single operation. From 1-8 memory modules may thus be shared by a maximum of 4 devices as shown in Figure 5. The modular design of the memory interchange allows the use of only the device switching electronics which are associated with the active channels.

In order to permit two or more PB440's to share a common portion of their memory as described above, it is only necessary to connect the "remote memory"

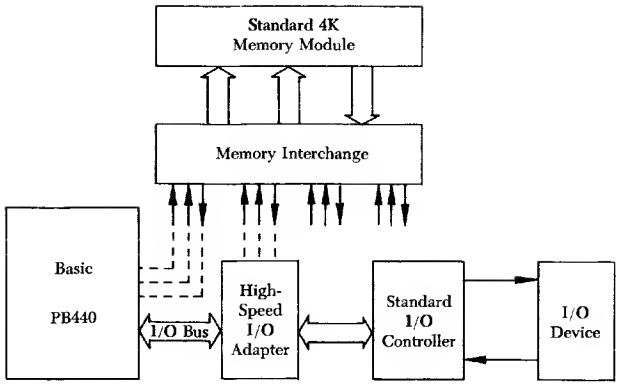


Figure 5. PB440 Shared Memory and High Speed I/O Options

output jacks of each of the individual computers to one set of the memory interchange input jacks. If the sharing device is instead an input-output controller (as illustrated in Figure 5), an additional unit, termed a high-speed I/O adapter, is required. During system assembly, an adapter is connected between the computer I/O bus and the controller for the selected device (a separate adapter is required for each device). The adapter transmits I/O commands from the computer to the controller and returns status responses to the computer as required by the basic I/O communications procedures.

In addition it contains a word counter and a current storage address counter which are filled by the program at the outset of a data block transfer between the device and the designated area of shared memory. The adapter thereafter functions to translate the controller's requests for inputs or outputs, into memory read or write signals, respectively, and conducts the corresponding data transmissions directly from or into memory. The computer may thus continue processing without interruption until such time as the desired number of words shall have been transferred. When the desired number of words have been transferred, the transmission is terminated, and if the option of requesting an interrupt has been exercised, the interrupt takes place at this point.

The maximum data transfer rates which can be accommodated by the above described use of the memory interchange and I/O adapter options are determined by the standard memory module cycle time (5 microseconds) and the degree of competition among the devices connected to a memory interchange. The resulting data rate capabilities are tabulated below for the conditions of 1-4 active channels. In addition to a 24-bit word rate, a character rate is quoted which is applicable to those character oriented devices whose I/O controller is equipped with an assembly-disassembly register for 6-bit characters. Also, the 24-bit word rate is quoted as a bit rate to facilitate comparison with competitive equipment.

SHARED MEMORY DATA RATES

Number of Simultaneously Active Channels	24-Bit Word Rate	6-Bit Char. Rate	Bit Rate
1	200 KC	800 KC	4.8 MC
2	100 KC	400 KC	2.4 MC
3	67 KC	268 KC	1.6 MC
4	50 KC	200 KC	1.2 MC

SUMMARY OF OPERATION CODES

All micro-steps are shown in the following tabulation. Symbols which appear in the "Operation" column are defined as follows:

SYMBOL	MEANING
[R]	Content of memory cell whose address is in R
n	Numeric Value
n_I	Numeric Value of Integer (at bit position 23)
n_X	Numeric Value of Integer (at bit position 8)
W	Working Storage Location
( )	Register Field
(S)	Sign Field of Register
(M)	Magnitude Field of Register
(X)	Exponent Field of Register
(F)	Fraction Field of Register
(Ch)	Character field, bits 18-23
(HN)	Bits 1-2 of N register
(HA)	High Address, bits 9-11
Rem	Remainder after division process
<N>	Repeat step until N = 0
(MD)	Mode field (bits 6-8) of "D" Register
K	Content of carry toggle 1 added to bit position 23
E X	Execute next micro-step (or Micro-Pair) if
N. C.	Carry toggles not set by "+" operation
→	Replaces
↔	Exchange contents
+	Add contents
*	Multiply
/	Divide
¬	"Not" or 1's complement
△	Logical Product
▽	Logical sum
⊕	Exclusive Or
$	Register Shift Operation
≡	Is identical with
[R]	As designated by convention assigned to R

Wherever the add operation (+) is indicated it can be assumed that the carry toggles will be set if the corresponding field takes part in the operation. Where a field does not take part, or where the symbol "N. C." is shown, the carry toggles are not affected.

The Roman numerals and letters which appear in the "Timing Group" column designate the group to which a micro-step is assigned for timing purposes. These groups are as follows:

GROUP	EXECUTION TIME
I	One clock pulse unless used as a right-half micro-step which replaces or alters the "P" register.
II	One clock pulse when used as a left-half micro-step; two clock pulses when used as a right-half micro-step.
III	The time depends upon the device addressed and whether the device is ready to return the status response or other information to the computer; the computer will delay until the status response is received. The normal execution time for a paper tape device which is ready is four clock pulses.
IV A	N clock pulses, where N is the number contained in the "N" register prior to execution of the group IV micro-step. The execution time may not, however, be zero. Therefore, if N is zero, the execution time is the same as if N were one, although the result will be that no register, except perhaps "P," is changed.
B	N clock pulses, where N is the number of bit positions shifted for normalization. A shift of zero bit positions and a shift of one bit position both require one clock pulse.
V	2N clock pulses, where N is the number contained in the "N" register prior to execution of the group V micro-step. As in group IV, the execution time is the same for N = O and N = 1.
VI A	Two o'clock pulses with a shadow time (see page 10 for a definition of shadow time) of three clock pulses when the operand is obtained from main memory by a left-half micro-step.
B	Three clock pulses with a shadow time of two clock pulses when the operand is obtained from main memory by a right-half micro-step.
C	The same time as for group II micro-steps (with no shadow time) when the operand is obtained from fast memory.
VII A	One clock pulse with a shadow time of four clock pulses when the operand is stored in main memory by a left-half micro-step.
B	Two clock pulses with a shadow time of three clock pulses when the operand is stored in main memory by a right-half micro-step.
C	Eight clock pulses with no shadow time when the operand is stored in fast memory by a left-half micro-step.
D	Nine clock pulses with no shadow time when the operand is stored in fast memory by a right-half micro-step.

Alphabetically by Mnemonic Code

MNEMONIC	OCTAL	MEANING	OPERATION	TIMING GROUP	PAGE
ADF R1 R2	41	Add Fractions	R1(F) + R2(F) → R2(F)	I	15
ADL R1 R2	55	Add Logical	R1 + R2 → R2	I	13
ADM R1 R2	43	Add Magnitudes	R1(M) + R2(M) → R2(M)	I	14
ADS R1 R2	73	Add Signs	R1(S) + R2(S) → R2(S)	I	14
ADX R1 R2	53	Add Exponents	R1(X) + R2(X) → R2(X)	I	15
AFK R1 R2	40	Add Fractions, Carry-in	R1(F) + R2(F) + K → R2(F)	I	15
ALC R1 R2	51	Add Logical for carry	R1 + R2	I	13
AMK R1 R2	42	Add Magnitudes, Carry-in	R1(M) + R2(M) + K → R2(M)	I	14
AND R1 R2	56	And (logical product)	R1 △ R2 → R2	I	13
BTR M	14	Backward Transfer Relative	¬0(¬ Ch) + M(Ch) + P → P; N. C.	II	21
CCF R1 R2	71	Copy Complement Fraction	¬ R1(F) → R2(F)	I	15
CCL R1 R2	63	Copy Complement Logical	¬ R1 → R2	I	12
CCM R1 R2	62	Copy Complement Magnitude	¬ R1(M) → R2(M)	I	14
CCS R1 R2	61	Copy Complement Sign	¬ R1(S) → R2(S)	I	14
CCX R1 R2	60	Copy Complement Exponent	¬ R1(X) → R2(X)	I	15
CDL R1 R2	54	Copy Decrement Logical	¬ 0 + R1 → R2; N. C.	I	13
CFS R1 R2	77	Copy From Special	[R1] → R2	I	16
CIL R1 R2	44	Copy, Increment Logical	R1 + 1 → R2	I	13
CIX R1 R2	52	Copy, Increment Exponent	R1(X) + 1_x → R2(X)	I	15
CLD M	07	Copy Literal to D	M → D(Ch); 0 → D(¬ Ch)	I	16
CLP M	04	Copy Literal to P	M → P(Ch); 7 → P(HA); 0 → P(¬ Ch ¬ HA)	II	20
COM R1 R2	65	Commutator control	Control com[R1]; Com ↔ R2[R1]	I	27
CPF R1 R2	70	Copy Fraction	R1(F) → R2(F)	I	15
CPL R1 R2	46	Copy Logical	R1 → R2	I	12
CPM R1 R2	47	Copy Magnitude	R1(M) → R2(M)	I	14
CPS R1 R2	72	Copy Sign	R1(S) → R2(S)	I	13
CPX R1 R2	50	Copy Exponent	R1(X) → R2(X)	I	15
CTS R1 R2	02	Copy To Special	R2 → [R1]	I	16
DTR R1 R2	10	Data Transfer	[R1] → R2 or R2 → [R1]	III	28
DVS R1 A	17	Divide Step	A, B(M)/R1 → B(M); Rem A(M) <N>	V	20
EMP R2	06	Execute Micro-Pair	[R2] → E	I	23
EXC R1 R2	45	Exchange	R1 ↔ R2	I	13
FTR M	16	Forward Transfer	P + M_i → P; N. C.	II	20
HLT	01	Halt	Halt	I	23

Alphabetically by Mnemonic Code

MNEMONIC	OCTAL	MEANING	OPERATION	TIMING GROUP	PAGE
LDI R1 R2	26	Load and Increment	[R1] → R2; R1 + 1_i → R1; N. C.	VI	11
LDM R1 R2	27	Load from Memory	[R1] → R2	VI	11
LDS R1 R2	22	Load Special	[R1] → R2	VI	12
LDW n R2	23	Load from Working	W_n → R2	VI	11
LOR R1 R2	64	Logical "Or"	R1 ▽ R2 → R2	I	13
LRC M	05	Load Repeat Count	M → N(X); O → (HN)	I	16
MPS R1 A	15	Multiply Step	B(M) * R1(M) + A(M) → A, B(M) <N>	IV A	20
NOP	00	No Operation		I	23
SDL R1	13	Shift Double Length	A, B $[R1] <N>	IV A	17
SEL R1 R2	12	Select	Status → RI; select [R2]	III	27
SFR	11	Shift Floating Right	A(F), B(M) $ right <N>	IV A	17
SL6 R1 R2	67	Shift Left Six	R1 $ left 6 → R2	I	13
SLC R1 A	03	Shift Left and Count	A[R1], B(M) $ left [R1] until [R1] = 1	IV B	18
SSL R1 R2	66	Shift Single Length	R2 $ [R1] <N>	IV A	17
STI R1 R2	24	Store and Increment	R2 → [R1]; R1 + 1_i → R1; N. C.	VII	11
STM R1 R2	25	Store into Memory	R2 → [R1]	VII	11
STF R1 R2	34	Set Toggle False	0 → [R1] [R2]	I	23
STS R1 R2	20	Store Special	R2 → [R1]	VII	12
STT R1 R2	36	Set Toggle True	1 → [R1] [R2]	I	23
STW n R2	21	Store into Working	R2 → W_n	VII	11
TCF R1 R2	35	Test Condition False	EX[R1] [R2] = 0	II	23
TCT R1 R2	37	Test Condition True	EX[R1] [R2] = 1	II	23
TMF R1 R2	31	Test Mode False	EX R1¬ ≡ (MD) [R2]	II	22
TMT R1 R2	33	Test Mode True	EX R1 ≡ (MD) [R2]	II	22
TNZ R1 R2	30	Test Non-Zero	EX R1¬ ≡ 0 [R2]	II	21
TZO R1 R2	32	Test for Zero	EX R1 ≡ 0 [R2]	II	21
XOR R1 R2	57	Exclusive Or	R1 ⊕ R2 → R2	I	13

Numerically by Octal Code

OCTAL	MNEMONIC	MEANING	OPERATION	TIMING GROUP	PAGE
00	NOP	No Operation		I	23
01	HLT	Halt	Halt	I	23
02	CTS R1 R2	Copy to Special	R2 → [R1]	I	16
03	SLC R1 A	Shift Left and Count	A[R1], B(M) $ left [R1] until [R1] = 1	IV B	18
04	CLP M	Copy Literal to P	M → P(Ch); 7 → P(HA); 0 → P(¬ Ch ¬HA)	II	20
05	LRC M	Load Repeat Count	M → N(X); O → (HN)	I	16
06	EMP R2	Execute Micro-Pair	[R2] → E	I	23
07	CLD M	Copy Literal to D	M → D(Ch);0 → D(¬ Ch)	I	16
10	DTR R1 R2	Data Transfer	[R1] → R2 or R2 → [R1]	III	28
11	SFR	Shift Floating Right	A(F), B(M) $ right <N>	IV A	17
12	SEL R1 R2	Select	Status →RI; select [R2]	III	27
13	SDL R1	Shift Double Length	A, B $[R1] <N>	IV A	17
14	BTR M	Backward Transfer Relative	¬ 0(¬ Ch) + M(Ch) + P→ P; N. C.	II	21
15	MPS R1 A	Multiply Step	B(M) * R1(M) + A(M) → A, B(M) <N>	IV A	20
16	FTR M	Forward Transfer	P + Mi → P; N. C.	II	20
17	DVS R1 A	Divide Step	A, B(M)/R1 → B(M), Rem A(M) <N>	V	20
20	STS R1 R2	Store Special	R2 → [R1]	VII	12
21	STW n R2	Store into Working	R2 → Wn	VII	11
22	LDS R1 R2	Load Special	[R1] → R2	VI	12
23	LDW n R2	Load from Working	Wn → R2	VI	11
24	STI R1 R2	Store and Increment	R2→ [R1]; R1 + li → R1; N. C.	VII	11
25	STM R1 R2	Store into Memory	R2 → [R1]	VII	11
26	LDI R1 R2	Load and Increment	[R1] → R2; R1 + li → R1; N. C.	VI	11
27	LDM R1 R2	Load from Memory	[R1] → R2	VI	11
30	TNZ R1 R2	Test Non-Zero	EX R1 ¬ ≡ 0[R2]	II	21
31	TMF R1 R2	Test Mode False	EX R1¬ ≡ (MD) [R2]	II	22
32	TZO R1 R2	Test for Zero	EX R1 ≡ 0 [R2]	II	21
33	TMT R1 R2	Test Mode True	EX R1 ≡ (MD) [R2]	II	22
34	STF R1 R2	Set Toggle False	0 → [R1] [R2]	I	23
35	TCF R1 R2	Test Condition False	EX[R1] [R2] = 0	II	23
36	STT R1 R2	Set Toggle True	1 → [R1] [R2]	I	23

Numerically by Octal Code

OCTAL	MNEMONIC	MEANING	OPERATION	TIMING GROUP	PAGE
37	TCT R1 R2	Test Condition True	EX[R1][R2] = 1	II	23
40	AFK R1 R2	Add Fractions, Carry-In	R1(F) + R2(F) + K → R2(F)	I	15
41	ADF R1 R2	Add Fractions	R1(F) + R2(F) → R2(F)	I	15
42	AMK R1 R2	Add Magnitudes, Carry-In	R1(M) + R2(M) = K → R2(M)	I	14
43	ADM R1 R2	Add Magnitudes	R1(M) + R2(M) → R2(M)	I	14
44	CIL R1 R2	Copy, Increment Logical	R1 + 1 → R2	I	13
45	EXC R1 R2	Exchange	R1 ↔ R2	I	13
46	CPL R1 R2	Copy Logical	R1 → R2	I	12
47	CPM R1 R2	Copy Magnitude	R1(M) → R2(M)	I	14
50	CPX R1 R2	Copy Exponent	R1(X) → R2(X)	I	15
51	ALC R1 R2	Add Logical for Carry	R1 + R2	I	13
52	CIX R1 R2	Copy, Increment Exponent	R1(X) + 1x → R2(X)	I	15
53	ADX R1 R2	Add Exponents	R1(X) + R2(X) → R2(X)	I	15
54	CDL R1 R2	Copy Decrement Logical	¬ 0 + R1 → R2; N. C.	I	13
55	ADL R1 R2	Add Logical	R1 + R2 → R2	I	13
56	AND R1 R2	And (logical product)	R1 △ R2 → R2	I	13
57	XOR R1 R2	Exclusive Or	R1 ⊕ R2 → R2	I	13
60	CCX R1 R2	Copy Complement Exponent	¬ R1(X) → R2(X)	I	15
61	CCS R1 R2	Copy Complement Sign	¬ R1(S) → R2(S)	I	14
62	CCM R1 R2	Copy Complement Magnitude	¬ R1(M) → R2(M)	I	14
63	CCL R1 R2	Copy Complement Logical	¬ R1 → R2	I	12
64	LOR R1 R2	Logical "Or"	R1 △ R2 → R2	I	13
65	COM R1 R2	Commutator Control	Control com [R1]; Com ↔ R2 [R1]	I	27
66	SSL R1 R2	Shift Single Length	R2 $ [R1] <N>	IV A	17
67	SL6 R1 R2	Shift Left Six	R1 $ left 6 → R2	I	13
70	CPF R1 R2	Copy Fraction	R1(F) → R2(F)	I	15
71	CCF R1 R2	Copy Complement Fraction	¬ R1(F) → R2(F)	I	15
72	CPS R1 R2	Copy Sign	R1(S) → R2(S)	I	13
73	ADS R1 R2	Add Signs	R1(S) + R2(S) → R2(S)	I	14
77	CFS R1 R2	Copy From Special	[R1] → R2	I	16

PB440 CHARACTER CODES

Binary	Octal	Character	Binary	Octal	Character
000000	00	0	100000	40	—
000001	01	1	100001	41	J
000010	02	2	100010	42	K
000011	03	3	100011	43	L
000100	04	4	100100	44	M
000101	05	5	100101	45	N
000110	06	6	100110	46	O
000111	07	7	100111	47	P
001000	10	8	101000	50	Q
001001	11	9	101001	51	R
001010	12	(BS)	101010	52	(CR)
001011	13	=	101011	53	$
001100	14	'	101100	54	*
001101	15	:	101101	55	]
001110	16	>	101110	56	;
001111	17	(LF)	101111	57	△
010000	20	+	110000	60	(SP)
010001	21	A	110001	61	/
010010	22	B	110010	62	S
010011	23	C	110011	63	T
010100	24	D	110100	64	U
010101	25	E	110101	65	V
010110	26	F	110110	66	W
010111	27	G	110111	67	X
011000	30	H	111000	70	Y
011001	31	I	111001	71	Z
011010	32	?	111010	72	≠
011011	33	.	111011	73	,
011100	34	)	111100	74	(
011101	35	[	111101	75	∽
011110	36	<	111110	76	(TAB)
011111	37	‡	111111	77	⧺

(BS)—Backspace
(LF)—Line Feed
(CR)—Carriage Return
(TAB)—Tabulate
(SP)—Space

pb Packard Bell Computer

A DIVISION OF PACKARD BELL ELECTRONICS

1905 ARMACOST AVENUE
LOS ANGELES 25, CALIFORNIA
GRanite 8-0051 • BRadshaw 2-9161